Freescale Semiconductor
Application Note

Document Number: AN3870
Rev. 0, 08/2010

Developing an Application for the i.MX Devices on the Linux Platform

by *Multimedia Applications Division*
Freescale Semiconductor, Inc.
Austin, TX

1 Introduction

This application note describes how to set up a Linux software development environment on the i.MX devices. The application note helps the user to cross-compile, deploy, and debug code for an i.MX device (with the GNU-ARM toolchains that are included in the Board Support Package—BSP) using the Eclipse Integrated Development Environment (IDE).

Contents

freescale semiconductor

The application note assumes that the user can boot Linux (one of the BSPs) on the target board through the Network File System (NFS) from a Linux host.

NOTE

The steps that are given in this application note are illustrated in the Fedora distribution that uses the Eclipse IDE, GNU Debugger (GDB), and i.MX31 Product Development Kit (PDK) board. However, these steps should be applicable to most of the Linux distributions (for the host) and i.MX devices with little or no change.

2 Prerequisites

The following are the prerequisites for developing an application for the i.MX devices on Linux platform:

- Linux distribution installed in the host computer
- Required services—Trivial File Transfer Protocol (TFTP), NFS, and serial communication—that are configured and running on the host computer
- i.MX Linux BSP running on the target board
- Familiarity with the Linux Target Image Builder (LTIB) configuration screen navigation and knowledge of the location where the common packages are selected or deselected

If the user requires any help to set up the prerequisites, refer to the Linux BSP documents. For example,for setting up the i.MX31 PDK, refer to the *i.MX31 PDK 1.5 Linux User's Guide* (926-77208), which is available in the BSP tarball that resides in the `/doc/` directory.

For the cross development, it is recommended to work on a root file system that is deployed through NFS. Any change to the target `rootfs` files in the Linux host can be detected immediately by the target, without any flash programming. If this is not possible on the user environment, establish a network connection between the i.MX board and host and use the TFTP or other services to transfer files between them.

3 Installing Eclipse

C/C++ developers should download and install the Linux version of the Eclipse IDE which is available at http://www.eclipse.org/downloads/. The latest Eclipse version for Linux is Eclipse 3.4 (also known as Ganymede and SR1). The Eclipse platform is delivered to the user as a tarball. The tarball contents can be decompressed to a directory inside the home page. The application can be started from the command line or by clicking the corresponding icon in the directory.

The commands to start the Eclipse application from the command line are as follows:

```
$ cp /<your_download_location>/eclipse-cpp-ganymede-SR1-linux-gtk.tar.gz /home/<your_user>/
$ cd /home/<your_user>/
$ tar xzvf eclipse-cpp-ganymede-SR1-linux-gtk.tar.gz
$ cd eclipse
$ ./eclipse
```

4 Configuring Cross Compilation and Deployment Using Preinstalled Toolchain

The steps to configure the cross compilation and deployment using the preinstalled toolchain are as follows:

1. The toolchains are installed by LTIB in a common path, `/opt/freescale/usr/local`, in the host machine. In this common path, a directory is available for every toolchain that have been previously installed as part of a BSP. Select the toolchain for the i.MX device and BSP. For example, to work with the i.MX31 PDK Linux Software Development Kit (SDK) 1.4, the user should use the `gcc-4.1.2-glibc-2.5-nptl-3` toolchain.

NOTE

At some point, the Linux kernel package of the BSP is required to be extracted so that the Eclipse can point to the contents of the Eclipse include directory. To extract the Linux kernel package, use the command, `./ltib -m prep -p kernel`, in the LTIB installation directory.

2. When IDE is run for the first time, select a directory for its workspace so that the projects can be stored. The default directory can be used for this purpose. The Use this as the default... box is not required to be checked. After selecting the workspace directory, the IDE finishes loading and displays a welcome tab. Close the welcome tab to view the default Eclipse perspective (work area layout).
3. Create a new C project of type executable as shown in Figure 1.

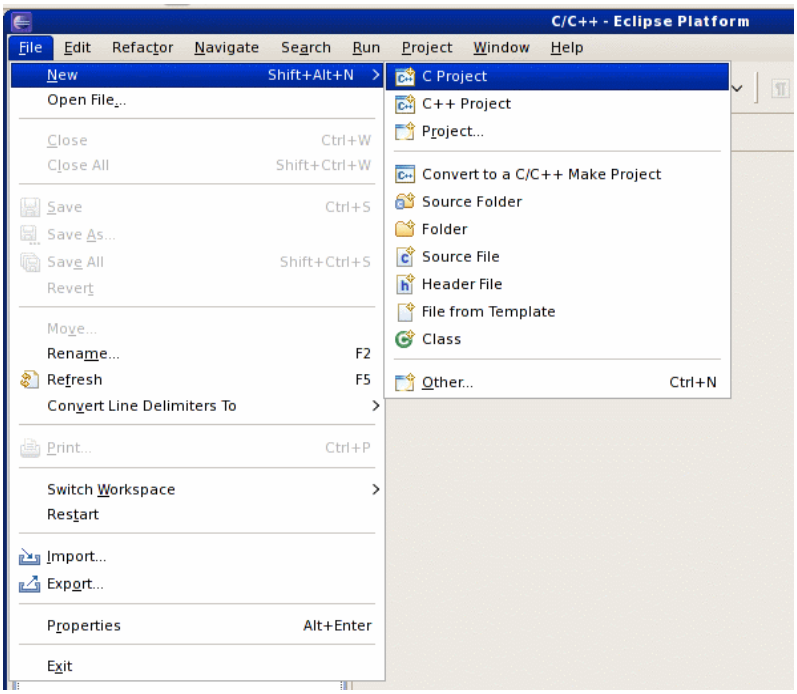

Figure 1. Creating a New C Project

4. Select the Hello World ANSI C template, and name the project uart_appnote as shown in Figure 2.

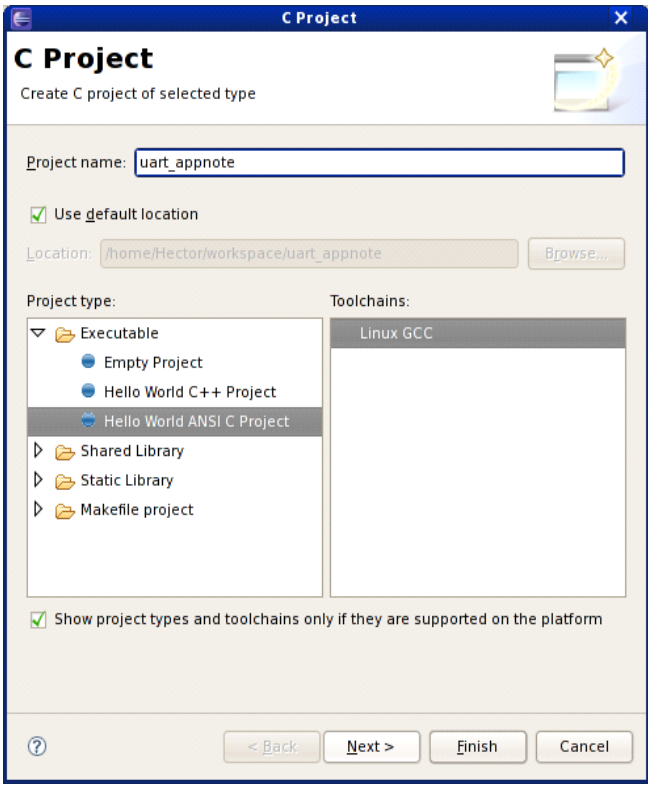

Figure 2. New Project Settings

5. For the toolchain settings, select Linux GCC as shown in Figure 2 (later, this is modified manually). Click Next and enter the user name and copyright notice. Click Next again and ensure that the two default configurations (Debug and Release) are checked. Once these options are set, click Finish.
6. Open the project properties by pressing Alt + Enter or by right clicking the project name that is located on the left side of the pane (the Project Explorer) and selecting the Properties option.

7. On the Properties window, click the C/C++ Build option that is located in the left side of pane. Click Manage Configurations and create a new build configuration as shown in Figure 3 with the following properties:
   - Name—MX31-Debug
   - Description—Cross-compile for i.MX31
   - Select Existing configuration option from the Copy settings from option list and select Debug

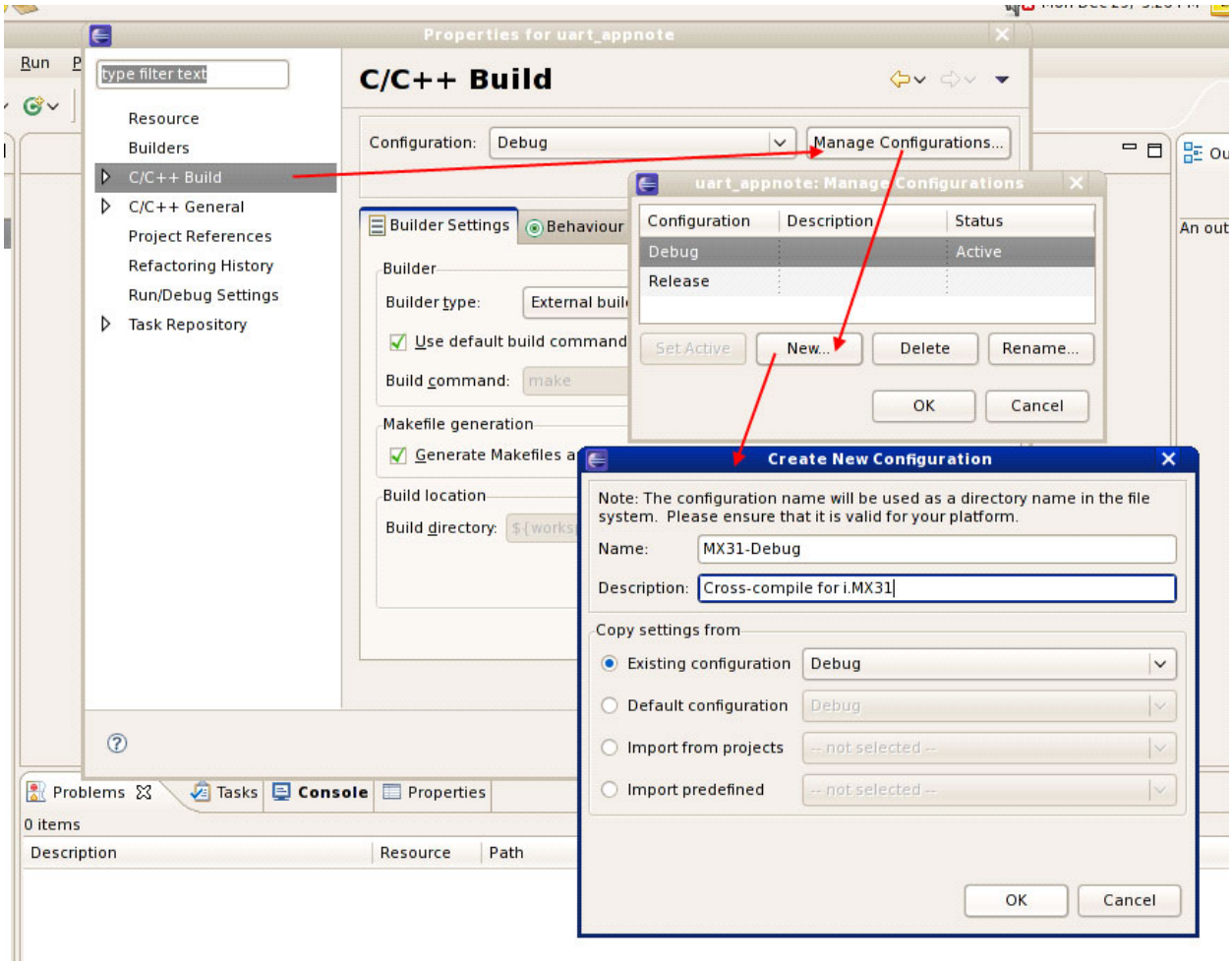

Figure 3. New Build Configuration

Click OK to close the Create New Configuration window.

8. Activate the new configuration on the Manage Configurations window by clicking the new configuration entry and then clicking Set Active. Click OK to close the Manage Configurations window.
9. Select the new configuration from the Configuration drop-down list in the C/C++ Build tab and click Apply.

10. Expand the C/C++ Build options and click Settings. On the Tool Settings tab, which is located on the right side of the pane, change the GCC C Compiler command from `gcc` to `/opt/freescale/usr/local/gcc-4.1.2-glibc-2.5-nptl-3/arm-none-linux-gnueabi/bin/arm-none-linux-gnueabi-gcc` as shown in Figure 4.

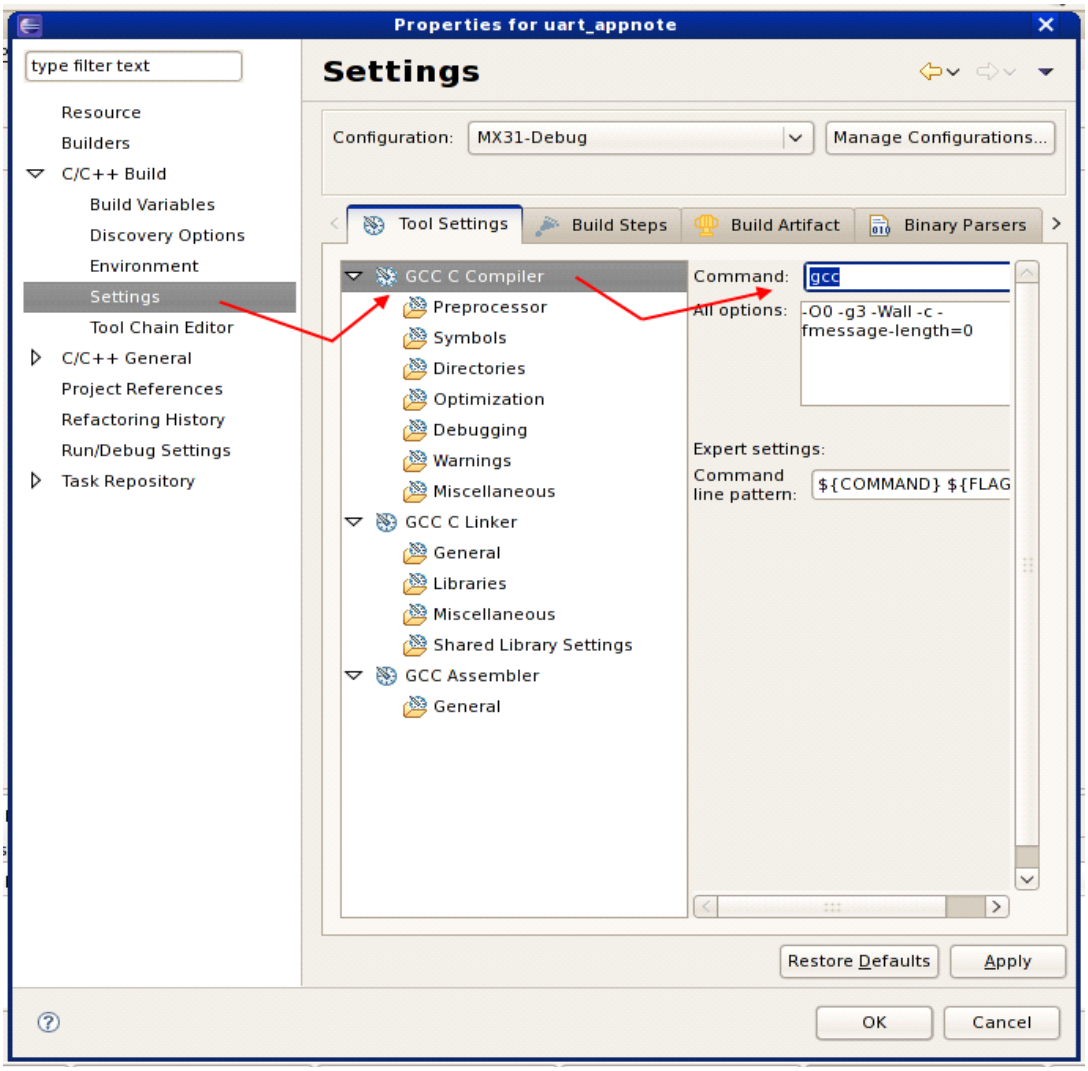

Figure 4. Replacing the gcc Command

11. Perform the following steps to add the `/opt/freescale/usr/local/gcc-4.1.2-glibc-2.5-nptl-3/arm-none-linux-gnueabi/bin/arm-none-linux-gnueabi-gcc` path to the include directory of the kernel:
    a) Click Directories which is located below the GCC C Compiler Tool Settings
    b) Click Add directory path
    c) On the Add directory path window, click File system and navigate to the path, `/<LTIB_PATH>/rpm/BUILD/linux/include`

d) Select the include directory and click OK as shown in Figure 5

NOTE

`/<LTIB_PATH>/` is the directory where the LTIB for the BSP is installed.

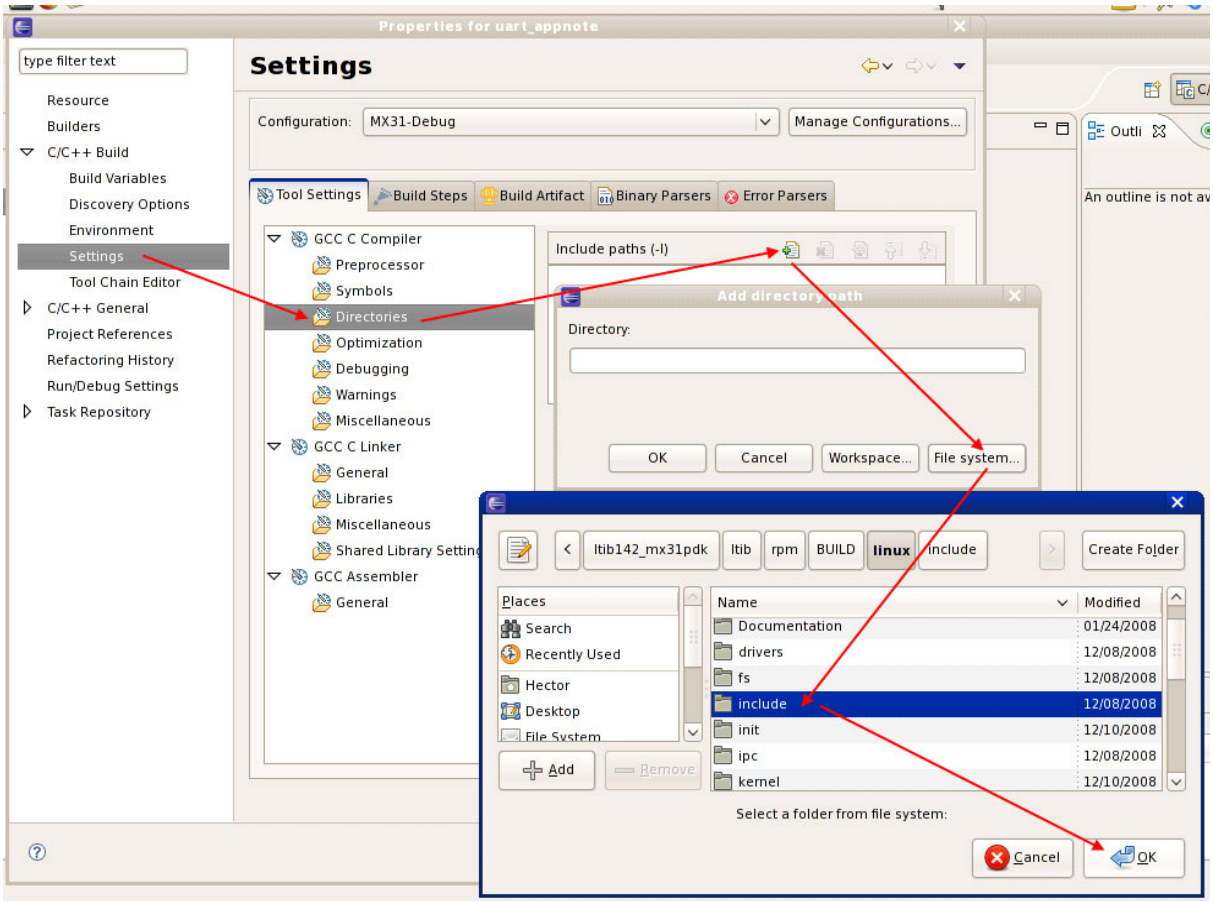

Figure 5. Adding the Include Directory

To add more paths to the include directory, repeat the process mentioned in this step.

12. Select GCC C Compiler > Optimization > Optimization Level and select None from the Optimization Level drop-down list (generally, this option is already set to the desired value).
13. Select GCC C Compiler > Debugging > Debug Level and select the Maximum option from the Debug Level drop-down list (generally, this option is already set to the desired value).
14. Follow the procedure in Step 10 and provide the appropriate values for GCC C Linker to modify the GCC C Linker command from `gcc` to `/opt/freescale/usr/local/gcc-4.1.2-glibc-2.5-nptl-3/arm-none-linux-gnueabi/bin/arm-none-linux-gnueabi-gcc`.
15. The linker should know the location of the linking libraries for the target root file system. To do this, add the `/<LTIB_PATH>/rootfs/lib` path to the libraries option in the GCC C Linker > Libraries > Library search path box.

16. Repeat the procedure mentioned in Step 15 to add the `/<LTIB_PATH>/rootfs/usr/lib` path to libraries.
17. Repeat the procedure described in Step 10 and Step 14 to modify the command for the GCC Assembler (located near the bottom of the Tool Settings tab). Here, the GCC Assembler command is changed from `as` to `/opt/freescale/usr/local/gcc-4.1.2-glibc-2.5-nptl-3/arm-none-linux-gnueabi/bin/arm-none-linux-gnueabi-as`.
18. Click the Build Steps tab and configure a simple deployment operation. Copy the built binary to the root file system that is exported to the target using NFS and configure Post-Build Steps with the following command:
    `cp /<workspace/project>/<output binary>  /<LTIB_PATH>/rootfs/home/`

    NOTE

    If Eclipse is not run as the root, change the permissions of the home directory of the target rootfs file in advance. In this case, Eclipse can write to the home directory during deployment.

    Now, the i.MX31 application is built and copied to `rootfs`. The application can be executed from the target and debugged using Eclipse.
19. Click the Binary Parsers tab. Perform the following steps to configure the binary parsers for Eclipse:
    a) Uncheck the Elf Parser and check the GNU Elf Parser checkbox
    b) On the Binary Parsers options of the GNU Elf Parser, change the addr2line Command from `addr2line` to `/opt/freescale/usr/local/gcc-4.1.2-glibc-2.5-nptl-3/arm-none-linux-gnueabi/bin/arm-none-linux-gnueabi-addr2line`
    c) Change the c++filt Command from `c++filt` to `/opt/freescale/usr/local/gcc-4.1.2-glibc-2.5-nptl-3/arm-none-linux-gnueabi/bin/arm-none-linux-gnueabi-c++filt`
    d) Click OK to close the Properties window

Now, the user should be able to build code and execute output applications on the target platform.

NOTE

By default, Eclipse is configured to build code for the x86 platforms. However, with this new configuration, user can generate ARM binaries that can run on the i.MX platform.

5 Configuring Cross Debugging with GDB

This section describes how to configure the cross debugging with the GDB debugger package. Before executing the configuration steps, ensure that the following GDB packages are selected on the LTIB configuration screen:

- GDB to run natively on the target
- Cross GDB (runs on the build machine)

- Extensible Markup Language (XML) support
- GDB server to run natively on the target

After the GDB packages are selected, exit from LTIB configuration screen and allow the build to complete. Ensure that the host computer has access to the internet so that the packages can be downloaded from the Global Package Pool (GPP).

The steps to create a debug configuration are as follows:

1. Click the downward arrow on the right of the bug icon to open the Debug Configurations window. Double click the C/C++ Local Application option to show a debug configuration for the project. Right click on this configuration and select Duplicate as shown in Figure 6.

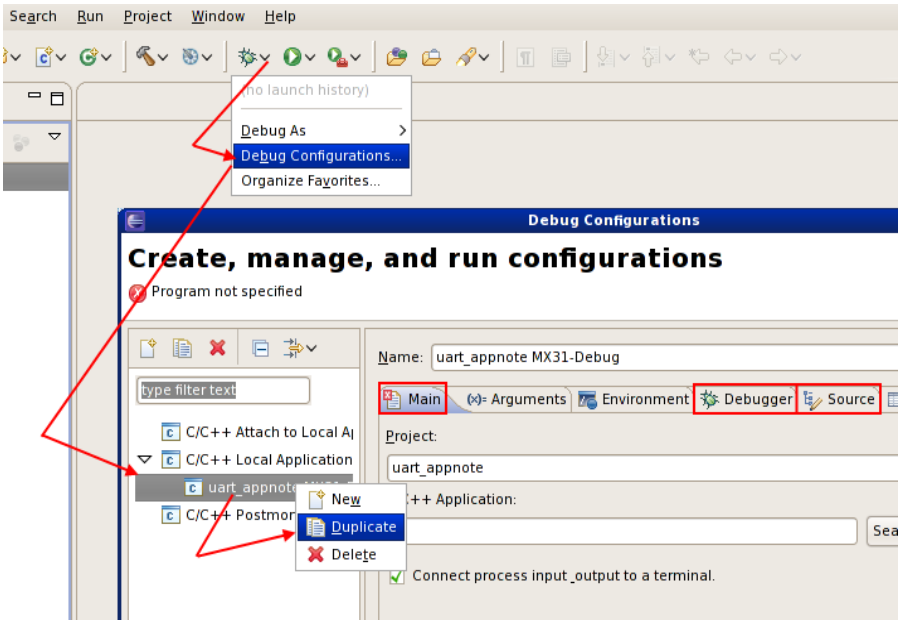

Figure 6. Creating a Debug Configuration

2. Select the following on the Main tab to configure the new debug configuration:
   - Name—i.MX Application Debugging-GDBServer
   - Project—<the created project>
   - C/C++ Application—<the ARM binary built with the toolchain> (browse to locate the ARM binary that resides in the Eclipse workspace)
3. Click the Debugger tab and perform the following:
   - Select the GDBServer debugger from the Debugger drop-down list
   - In Debugger Options, use the ARM capable debugger that is built by LTIB:
     - GDB debugger—`/<LTIB_PATH>/bin/gdb`
     - GDB command set—Standard
     - Protocol—mi

— On the Shared Libraries tab (under the Debugger options), add the following shared library paths:
  - `/<LTIB_PATH>/rootfs/lib`
  - `/<LTIB_PATH>/rootfs/usr/lib`

4. Configure the connection on the Connection tab with the following attributes:
   — Type—TCP
   — Host name or IP address—192.168.0.2 (use the IP address of the target i.MX board)
   — Port number—10000
5. Ensure that the project folder is listed on the Source tab
6. Close the Debug Configurations window

Now, the user can start a cross-debugging session with the Eclipse and i.MX device.

6 Writing or Editing, Building, and Deploying an Application

The steps to build code and copy the output binary to the target rootfs home directory are as follows:

1. The Universal Asynchronous Receiver/Transmitter (UART) unit test is used as the example application. Therefore, the UART unit test source files must be decompressed. This can be done by issuing the following commands on the host:

```
$ cd /<LTIB_PATH>/
$ ./ltib -m prep -p imx-test
```

   These commands extract the source files into the `rpm/BUILD/` folder.

2. Replace the contents of the Eclipse project source file with the contents of the UART unit test .c source file or overwrite the file with the following command:

```
$ cp /<LTIB_PATH>/rpm/BUILD/imx-test2.3.2/test/mxc_uart_test/mxc_uart_test.c
/<ECLIPSE_WORKSPACE>/uart_appnote/src/uart_appnote.c
```

3. On Eclipse Project Explorer, expand the `src` folder and double click the `uart_appnote.c` file to open the source file on the editor window. Now, replace line 33, `printf("Test: MXC UART!\n");`, with `printf("Test: MXC UART with Eclipse!\n");`. Save and build the project by clicking the floppy disk and hammer icons, respectively as shown in Figure 7.

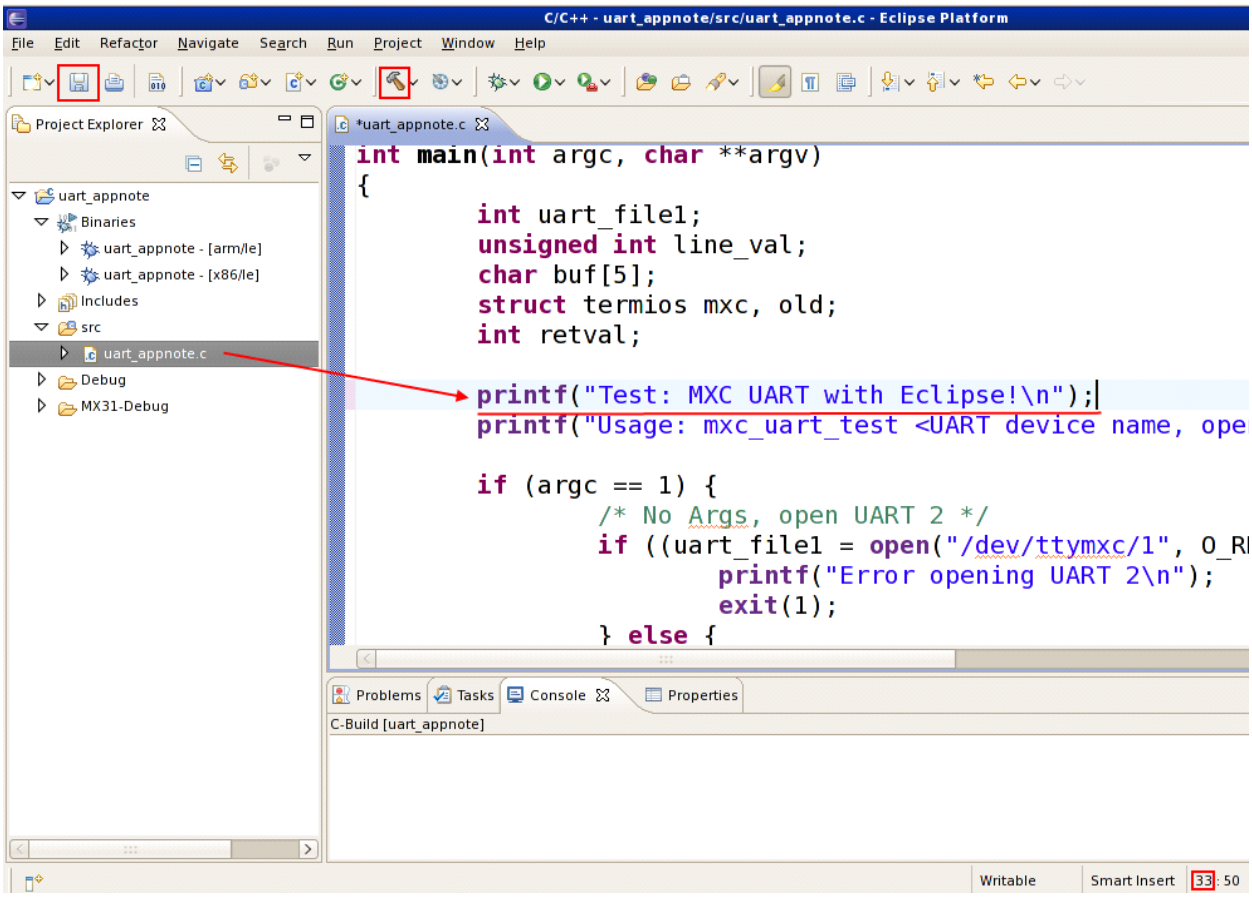

Figure 7. Modifying the Source Code

Now, the user should be able to build the code and copy the output binary to the home directory of the target rootfs file. The application can be tested by entering the following commands on the target serial terminal emulator window:

```
$ cd /home/
$ ./uart_appnote /dev/ttymxc2
```

If the code is built properly and the output directory is copied correctly, then the output of the unit test is as follows:

```
Test: MXC UART!
Usage: mxc_uart_test <UART device name, opens UART2 if no dev name is specified>
/dev/ttymxc2 opened
Attributes set
Test: IOCTL Set
Data Written= Test
Data Read back= Test
```

The test then opens the UART port that is specified in the command line. If the UART port is not specified in the command line, then the test opens the UART2 port (zero-indexed is ttymxc1). After opening the UART port, the program sets the loopback feature on the line discipline and the data is written to or read from the UART port.

The serial test completes with no errors (returns code 0) if everything works properly. If there are errors while opening the UART port or during the I/O operations, the program exits with a different error code or hang. In this case, the problem can be debugged with the Eclipse debug environment by observing the variables, structures, breakpoint additions, and so on.

7 Debugging

Steps to perform the debug operation are as follows:

1. Execute the following command on the target terminal at the path where the test application is located:

   ```
   $ gdbserver *:10000 uart_appnote
   ```

   When this command is executed, the following is displayed on the screen:

   ```
   Process uart_appnote created; pid = 1799
   Listening on port 10000
   ```

2. As the target is ready, click the Bug icon on the Eclipse Platform window and select the GDBServer debug configuration that is created in Section 5, “Configuring Cross Debugging with GDB,” to attach the debugger. If the host and i.MX platforms are connected successfully, then the user is asked to confirm the perspective switch. After the confirmation, the GDB server displays the message, `Remote debugging from host 192.168.0.1`, on the target terminal and Eclipse opens the debug perspective with the source code, disassembly, variables, registers, and console output.

3. Check the Remember my decision box and click Yes (the user can switch between the perspectives by clicking the corresponding icons that are located on the top right corner of the Eclipse Platform window as shown in Figure 8).

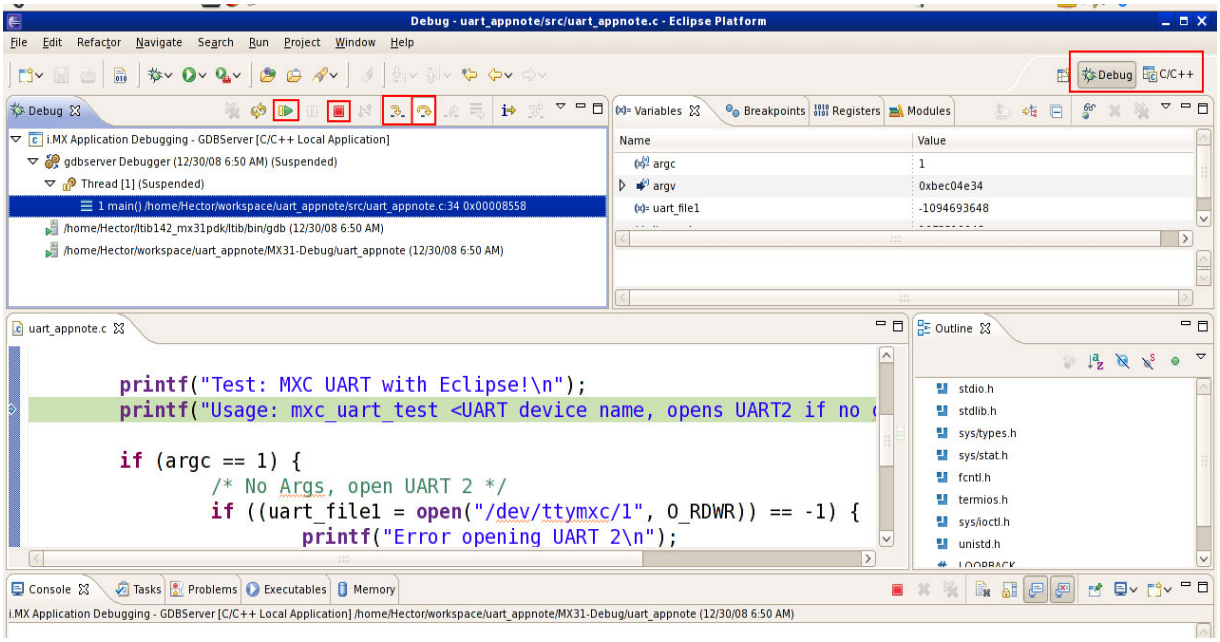

Figure 8. Eclipse Debugging Perspective

As the glibc functions (which are accessed from a shared lib binary) are used, these are required to be navigated by stepping over them (by clicking the curved yellow arrow icon shown in Figure 8 or by pressing F6). The user is now only able to step into the functions that are part of their application (by clicking the straight-angle yellow arrow icon shown in Figure 8 or by pressing F5) or applications for which the source code is available.

Important points that are helpful for performing the debugging operation are as follows:

- The program execution can be resumed by clicking the green play icon highlighted in Figure 8 or by pressing F8.
- The debugging session can be stopped with the red stop button highlighted in Figure 8 or by pressing Ctrl + F2 (this kills the processes associated with the debugging session on the target and host).
- If the initial parameters are required to be passed to the application that is being debugged, then it can be done while launching the GDB server with the following command:

```
$ gdbserver *:10000 uart_appnote /dev/ttymxc2
```

The user is now able to cross-compile, deploy, and debug code for an i.MX device using the Eclipse IDE and GNU tools on the Linux platform.

8 Revision History

Table 1 provides a revision history for this application note.

Table 1. Document Revision History

Rev. Number	Date	Substantive Change(s)
0	08/2010	Initial release.

THIS PAGE INTENTIONALLY LEFT BLANK

How to Reach Us:

Home Page:
www.freescale.com

Web Support:
http://www.freescale.com/support

USA/Europe or Locations Not Listed:
Freescale Semiconductor, Inc.
Technical Information Center, EL516
2100 East Elliot Road
Tempe, Arizona 85284
1-800-521-6274 or
+1-480-768-2130
www.freescale.com/support

Europe, Middle East, and Africa:
Freescale Halbleiter Deutschland GmbH
Technical Information Center
Schatzbogen 7
81829 Muenchen, Germany
+44 1296 380 456 (English)
+46 8 52200080 (English)
+49 89 92103 559 (German)
+33 1 69 35 48 48 (French)
www.freescale.com/support

Japan:
Freescale Semiconductor Japan Ltd.
Headquarters
ARCO Tower 15F
1-8-1, Shimo-Meguro, Meguro-ku
Tokyo 153-0064
Japan
0120 191014 or
+81 3 5437 9125
support.japan@freescale.com

Asia/Pacific:
Freescale Semiconductor China Ltd.
Exchange Building 23F
No. 118 Jianguo Road
Chaoyang District
Beijing 100022
China
+86 10 5879 8000
support.asia@freescale.com

For Literature Requests Only:
Freescale Semiconductor
Literature Distribution Center
1-800 441-2447 or
+1-303-675-2140
Fax: +1-303-675-2150
LDCForFreescaleSemiconductor
@hibbertgroup.com

Document Number: AN3870
Rev. 0
08/2010